# Cyberbullying / Toxic Comment Detection Project

## 1. Introduction

This project focuses on detecting cyberbullying and toxic comments using a fine-tuned Language Learning Model (LLM).   
The system identifies harmful or inappropriate online content and classifies it into categories such as hate speech, harassment, or general toxicity.  
The project also includes a web-based user interface (UI) for users to input comments and receive real-time toxicity detection results.

## 2. Objectives

1. To develop an AI-based model capable of detecting and classifying toxic or bullying comments.  
2. To create a user-friendly web interface for testing and demonstration.  
3. To train and fine-tune an existing LLM (such as BERT, RoBERTa, or DistilBERT) on a dataset of labeled comments.  
4. To integrate the trained model with a web application backend (Flask or FastAPI).  
5. To display the prediction results clearly with confidence scores and category labels.

## 3. Dataset

Dataset Source: Kaggle’s “Toxic Comment Classification” dataset or similar publicly available datasets.  
Dataset Features:  
- id: Unique identifier for each comment.  
- comment\_text: The text of the comment.  
- toxic, severe\_toxic, obscene, threat, insult, identity\_hate: Binary labels for each type of toxicity.  
Data Preprocessing Steps:  
- Text cleaning (removing emojis, URLs, stopwords, and special characters)  
- Tokenization  
- Lemmatization  
- Handling class imbalance (e.g., SMOTE, undersampling, or weighting)

## 4. Model and Implementation

Model: Fine-tuned LLM such as BERT or RoBERTa.  
Implementation (50% weightage):  
- Load and preprocess the dataset.  
- Train-test split (typically 80-20).  
- Tokenize the text using the LLM tokenizer.  
- Fine-tune the pre-trained model on the toxic comment dataset.  
- Save the trained model (.pt or .pkl format).  
- Evaluate the model using accuracy, F1-score, precision, recall, and confusion matrix.  
- Export the final model for integration with the web app.

## 5. Backend (Flask / FastAPI)

Backend Implementation (25% weightage):  
- Use Flask or FastAPI as the backend framework.  
- Load the fine-tuned model at startup.  
- Create API endpoints (e.g., /predict) to accept user comments and return classification results.  
- Handle prediction requests asynchronously for faster response.  
- Return JSON response with the comment, predicted label(s), and confidence score(s).  
Example JSON output:  
{  
 "comment": "You're such a loser!",  
 "label": "toxic",  
 "confidence": 0.94  
}

## 6. Frontend (Web UI)

Frontend Implementation (25% weightage):  
- Create a simple and intuitive UI using HTML, CSS, and JavaScript (or React for advanced interfaces).  
- Components:  
 - Text input box for entering comments.  
 - "Analyze" button to send text to backend API.  
 - Result section to display toxicity result and confidence.  
- Optional color coding for results:  
 - Green: Safe  
 - Orange: Mildly Toxic  
 - Red: Highly Toxic

## 7. System Workflow

1. User enters a comment into the web interface.  
2. The comment is sent to the backend API.  
3. The backend sends the text to the trained model for analysis.  
4. The model predicts whether the comment is toxic or non-toxic (and specific categories if applicable).  
5. The backend returns the result to the frontend.  
6. The frontend displays the result with visual cues and confidence levels.

## 8. Technologies Used

- Python 3.x  
- Transformers (Hugging Face)  
- PyTorch or TensorFlow  
- Flask / FastAPI  
- HTML, CSS, JavaScript  
- Pandas, NumPy, Scikit-learn  
- Matplotlib / Seaborn (for analysis and visualization)

## 9. Future Enhancements

- Add sentiment-based severity scoring.  
- Integrate a "comment rewriting" feature to suggest polite alternatives.  
- Add multilingual support for detecting toxicity in other languages.  
- Enable batch file uploads for bulk comment analysis.  
- Deploy on cloud (e.g., AWS, Render, HuggingFace Spaces) for public use.

## 10. Conclusion

This project aims to leverage AI and Natural Language Processing to make online platforms safer by detecting and flagging toxic comments automatically.  
By combining the power of LLMs with a user-friendly web interface, this system demonstrates the practical use of machine learning in promoting healthy online communication.